

## REMARKS

Reconsideration of the above-identified patent application in view of the amendments above and the remarks following is respectfully requested.

Claims 10, 11, 13-15 and 19-22 are in this case. Claims 10, 11, 13-15 and 19-22 have been rejected under § 102(b). Claims 10, 11 and 14 have been rejected under § 102(e). Independent claims 10 and 15 have been amended. New claims 23-25 have been added.

The claims before the Examiner are directed toward a system and method for enabling an executing entity of a host system to directly execute code stored in a non-executable memory component. One or more executable memory components are provided to buffer a portion of the code to an executing entity for direct execution. The executable memory component(s) receive the code from the non-executable memory component via an interface separate from the standard memory interface that supports the direct execution of the code in the executable memory component(s). When more than one executable memory component is used, then while one executable memory component functions as a buffer of one portion of the code, another portion of the code is downloaded from the non-executable memory component to another executable memory component.

### § 102(b) Rejections - Microsoft

The Examiner has rejected claims 10, 11, 13-15 and 19-22 under § 102(b) as being anticipated by *Microsoft Windows 95 Resource Kit* (henceforth, "Microsoft"). The Examiner's rejection is respectfully traversed.

Microsoft teaches a computer that includes a non-executable memory component (hard disk), an executable memory component (RAM) and an executing

entity (processor). The RAM receives code from the hard disk and presents the code to the processor for direct execution. As best understood, the code is exchanged among the hard disk, the RAM and the processor on a common bus, as illustrated in US 6,434,695, which reference was submitted along with the response dated October 27, 2003.

Claim 10, as amended in the response dated October 27, 2003, recited the executable memory component as receiving the stored code "directly" from the non-executable memory component. The Examiner interprets the scope of "directly" to include via a bus such as the bus illustrated in US 6,434,695 that is shared by other components such as a processor. The Examiner appears to suggest that the recitation of a bus that is used exclusively by the executable memory component(s) and the non-executable memory component, and that is not shared by the processor, would overcome this rejection.

Therefore, independent claims 10 and 15 have been amended again. Claims 10 and 15 have been amended to state that the executing entity executes the code directly from the executable memory component(s) via a standard memory interface. Claim 10 has been amended to state that the executable memory component(s) receive(s) the stored code from the non-executable memory component via an interface that is separate from the standard memory interface. Claim 15 has been amended to state that the downloading of the code portions from the non-executable memory component to the executable memory component is via an interface separate from the standard memory interface.

Support for these amendments is found in the specification as follows. Page 1 lines 14-17 defines "direct" execution of code as including execution via a "standard memory interface":

The ability to execute code directly from a memory device usually requires the following characteristics from the memory device:

1. Standard memory interface, which includes address bus, data bus and a few more control signals (read enable, write enable etc.) (emphasis added)

Figure 2 has an arrow from NAND array 30 to executable buffer/s 20 to indicate the interface for downloading code from NAND array 30 to executable buffer/s 20. This interface is separate from the address bus and the data bus shown in the lower part of Figure 2. That this arrow indicates a feature of the hardware of the present invention, and is not merely a conceptual symbol to indicate data flow, is shown by page 10 line 11, which states that Figure 2 illustrates “architecture”.

Amended independent claims 10 and 15 now feature language which makes it absolutely clear that in the present invention, code is downloaded from the non-executable memory component to the executable memory component(s) via an interface that is separate from the standard memory interface that the executing entity uses to execute the code directly. Applicant believes that the amendment of the claims completely overcomes the Examiner's rejections on § 102(b) grounds.

With independent claims 10 and 15 allowable over Microsoft in their present form, it follows that claims 11, 13, 14 and 19-22, that depend therefrom, also are allowable over Microsoft.

Although claims 13 and 20-22 are allowable over Microsoft merely by virtue of depending from claims 10 and 15, Applicant takes the liberty of presenting another reason why these claims are allowable over Microsoft. The Examiner has cited memory paging, as described on pages 975-976 of Microsoft, as anticipating claims 13 and 20. The illustration on page 976 shows the mapping of two virtual memory spaces, of two different processes, into physical memory. This is not what is recited in claims 13 and 20. Claim 13 recites a plurality of executable memory components.

Claim 20 recites downloading a third portion of code to a second executable memory component. It is clear from the specification that the executable memory components are separate physical components. See, for example, page 10 line 19 through page 11 line 2:

A further preferred embodiment of the present invention describes a dual or multiple SRAM buffer 20. This buffer, or set of buffers, as can be seen in figure 2 can be used in order to prevent the memory from being locked for accesses during download operations. In this case the download operation will load the requested content to one SRAM buffer, referred to as download logic 22, while the other SRAM buffer 20 remains accessible and executable. The executable buffer 20 can be expanded to include two or more executable buffers. (emphasis added)

The two SRAM buffers clearly are separate physical entities. This is in contrast to the two virtual memory spaces illustrated on page 976 of Microsoft. As best understood, these two virtual memory spaces are mapped into different parts of the same physical RAM. There is neither a hint nor a suggestion in Microsoft, or in US 6,434,695 for that matter, of any utility to having more than one RAM.

To emphasize this distinction between the present invention and the prior art cited by the Examiner, new claims 23 and 24 have been added. New claims 23 and 24 state that the executable memory components recited in claims 13 and 20, respectively, are physically separate components. Support for new claims 23 and 24 is found in the specification in the above citation from page 10 line 19 through page 11 line 2.

#### **§ 102(e) Rejections – Hwang ‘399**

The Examiner has rejected claims 10, 11 and 14 under § 102(e) as being anticipated by Hwang, US Patent No. 6,263,399 (henceforth, “Hwang ‘399”). The Examiner’s rejection is respectfully traversed.

Hwang '399 teaches a memory interface 24 that acts as an intermediary between a CPU 10 and a NAND flash memory 26. As noted in column 6 line 23, memory interface 24 could include RAM; and this RAM could be used for direct execution of code from NAND flash memory 26 by CPU 10.

Claim 10, as amended in the response dated October 27, 2003, recites a feature of the present invention that is lacking in the teachings of Hwang '399: a mechanism for guaranteeing availability in an executable memory component of code requested by the executing entity. It is necessary, when the code requested by CPU 10 for execution is not yet stored in the RAM, to indicate to CPU 10 that CPU 10 needs to wait for the requested code to be downloaded from NAND flash memory 26. Hwang '399 is silent on this issue. Thus, the present invention is not anticipated from Hwang '399. Furthermore, the present invention is not even obvious from Hwang '399. The general thrust of Hwang '399 is towards the translation of signals from CPU 10 into a form that NAND flash memory 26 understands. This is stated *inter alia* in column 5 lines 20-24:

The key to the present invention is that memory interface 24 translates the communication protocol used by the CPU into a form usable by NAND flash memory 26 memory interface 24 communicates with both CPU 10 and NAND flash memory 26 in their native form.

There is neither a hint nor a suggestion in Hwang '399 of either the necessity of indicating to CPU 10 to wait for code to be downloaded from NAND flash memory 26 or of how to accomplish this.

These arguments were presented in the response dated October 27, 2003. In reply, the Examiner has interpreted "guaranteeing" the availability of code in memory interface 24 as being inherent in the teachings of Hwang '399. In the Examiner's words,

If there is no guarantee that the data stored in NAND memory can be made available to the CPU, Hwang's invention would be useless.

The Examiner appears to suggest that reciting a mechanism for indicating availability would overcome the rejection.

Therefore, claim 10 has been amended to recite a mechanism for indicating availability of code in one of the executable memory components. Support for this amendment is found in the specification in the paragraph on page 9 line 17 to page 10 line 2. In the response dated December 8, 2002, the first two lines of this paragraph were cited in support of (then new) claim 12, which introduced the limitation of "guaranteeing" code availability. In fact, the full paragraph describes the indication of code availability, using a busy signal. The full paragraph is as follows:

The device of the present invention manages at least one algorithm to guarantee availability of the requested information in the executable buffer/buffers. The device supplies a busy signal in cases when the information is not yet available. For example, in cases when the required data (according to the data address) is not within the SRAM current range, the data must be downloaded from the NAND to the SRAM. This operation takes time (download latency), during which the required addresses cannot return the required code (the required content). The provided busy signal therefore alerts the host system to cause the host system to cease data requests until downloading of the data is complete.

Amended independent claim 10 and 15 now feature language which makes it absolutely clear that the device of the present invention includes a mechanism for indicating availability, in one of the executable memory components, of code requested by the executing entity. Applicant believes that the amendment of the claims completely overcomes the Examiner's rejections on § 102(e) grounds.

With claim 10 allowable over Hwang '399 in its present form, it follows that claims 11 and 14, that depend therefrom, also are allowable over Hwang '399.

### Further Remarks

It has been demonstrated that claim 10, as now amended, is allowable over both Microsoft and Hwang '399 considered separately. The following remarks are directed at precluding a rejection of claim 10, as now amended, as an obvious combination of these two prior art references.

As noted above, Microsoft fails to teach or suggest the use of an interface, separate from the standard memory interface that the executing entity uses to execute code directly from the executable memory component, for sending code from the non-executable memory component to the executable memory component. Nevertheless, indicating the availability of requested code in the executable memory component could be construed as inherent in "process scheduling and multitasking" as described on pages 974-976 of Microsoft.

Also as noted above, Hwang '399 fails to teach or suggest indicating the availability of requested code in the executable memory component. Nevertheless, Hwang '399 appears to teach an interface (lines 28, 30, 32, 34 and 36) between memory interface 24 and CPU 10.

Lest the Examiner be tempted to reject claim 10, as now amended, as an obvious combination of the teachings of Microsoft and Hwang '399, Applicant respectfully reminds the Examiner of MPEP 706.02(j):

To establish a *prima facie* case of obviousness...there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings.

In this case, there is no such suggestion or motivation. As noted above, Microsoft has neither a hint nor a suggestion of any utility to the use of two separate interfaces to the executable memory component, one for downloading code from the non-executable memory component and the other for direct execution of the code by

the executing entity; and Hwang '399 is silent on the issue of indicating availability of code in memory interface 24 to be executed by CPU 10. The combination in the same device of an executable memory component that receives code from a non-executable memory component, for presentation for execution by an executing entity via a standard memory interface, via another interface separate from the standard memory interface, and a mechanism for indicating availability of the code to the executing entity, is not obvious from the prior art cited by the Examiner, either separately or in combination.

#### **Other New Claims**

New independent claim 25 has been added.

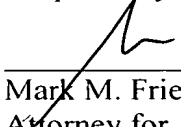
New independent claim 25 distinguishes the present invention from Microsoft somewhat differently than claim 10 as now amended. Specifically, new independent claim 25 states that the executable memory component receives the stored code from the non-executable memory component independently of the executing entity. This is in contrast to the computer systems addressed by Microsoft. As is well-known in the art, in such computer systems, the processor downloads the code to be executed from the hard disk to the RAM.

Support for new independent claim 25 is found in the specification in Figure 2, in which the arrow from NAND array 30 to executable buffers 20 indicates direct reception of stored code by executable buffers 20 from NAND array 30, without the intermediation of the executing entity via the address bus and the data bus. As noted above, that this arrow indicates a feature of the hardware of the present invention, and is not merely a conceptual symbol to indicate data flow, is shown by page 10 line 11, which states that Figure 2 illustrates "architecture".



In view of the above amendments and remarks it is respectfully submitted that independent claims 10 and 15, and hence dependent claims 11, 13, 14 and 19-24 are in condition for allowance. Prompt notice of allowance is respectfully and earnestly solicited.

Respectfully submitted,



---

Mark M. Friedman  
Attorney for Applicant  
Registration No. 33,883

Date: February 19, 2004